Friedrich-Alexander-Universität Erlangen-Nürnberg

Schriftliche Prüfung zur Vorlesung
“Einführung in die Programmierung für Ingenieure (C)”

Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Aufgabe 6	Gesamt
20	**20**	**10**	**10**	**10**	**10**	**80**

(Name) (Vorname) (Matrikel-Nr.)

(Fachrichtung) (Semester)

Erlangen, den 27.07.1996

Organisatorische Hinweise

Die folgenden Informationen bitte aufmerksam lesen und die Erklärung am Blattende unterschreiben.

- Die Bearbeitungszeit beträgt 120 Minuten.
- Es sind **keine** Hilfsmittel zugelassen.
- Schmierpapier darf **nicht** mit abgegeben werden. Die Lösung einer Aufgabe soll auf das Aufgabenblatt in den dafür vorgesehenen freien Raum geschrieben werden, der jeweils der Aufgabe folgt. Sollte der Platz nicht ausreichen, so können Sie die Rückseiten der Aufgabenblätter mitverwenden.
- Bei Bedarf sind von der Aufsicht Schmierpapier (farbig) und zusätzliche Lösungsblätter (weiß) zu erhalten. Vermerken Sie vor dessen Verwendung unbedingt Ihren Namen und Ihre Matrikelnummer darauf! **Schmierpapier darf nicht mit abgegeben werden!**
- Fragen zu den Prüfungsaufgaben können grundsätzlich nicht beantwortet werden.
- Tragen Sie Ihren Namen und Vornamen, Ihre Matrikelnummer, Studienfachrichtung und Fachsemesterzahl auf dem Deckblatt der Klausur ein.
- Bitte legen Sie Ihren Studenten- und einen Lichtbildausweis zur Kontrolle bereit.
- Sie dürfen den Raum nicht verlassen, bevor Ihre Personalien überprüft wurden und Sie die Klausurunterlagen der Aufsicht zurückgegeben haben.
- Bleiben Sie bitte am Ende der Bearbeitungszeit solange an Ihrem Platz sitzen, bis alle Klausurunterlagen eingesammelt sind.

Erklärung

Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen (10 Seiten inkl. Deckblatt).
- die Kenntnisnahme der obigen Informationen.

Ich bin damit einverstanden, daß mein Prüfungsergebnis unter Angabe der Matrikel-Nr. veröffentlicht wird.

☐ ja ☐ nein

Der Schein soll direkt an das Prüfungsamt geschickt werden:

☐ ja ☐ nein

Erlangen, den 27.07.1996

(Unterschrift)

Aufgabe 1: (20 Punkte)

Schreiben Sie eine Funktion zum Multiplizieren zweier Polynome. Die Polynome werden im Programm durch eine Struktur, bestehend aus dem Grad des Polynoms und einem Feld mit seinen Koeffizienten repräsentiert. Diese Struktur hat damit folgendes Aussehen:

```
struct poly_t {
        int         grad;
        double      *koeff;
};
```

So hätte etwa für das Polynom $p(x) = 1 + 5x + 3.3x^2 + 2x^3$ ***grad*** den Wert 3 und ***koeff*** wäre ein Zeiger auf ein Feld der Länge 4 mit den Werten 1, 5, 3.3 und 2.

Die von Ihnen zu erstellende Funktion

```
struct poly_t poly_mult(struct poly_t a, struct poly_t b)
```

erhält die Strukturen zweier Polynome und liefert als Resultat wieder eine derartige Struktur, wobei der Speicher für die darin enthaltenen Koeffizienten dynamisch anzufordern ist!

Die Multiplikation ist dabei nach folgender Formel durchzuführen:

Die Koeffizienten des Ergebnispolynoms

$$P_c(x) = \sum_{i=0}^{n+m} c_i x^i$$

ergeben sich aus den Polynomen

$$P_a(x) = \sum_{i=0}^{m} a_i x^i \text{ und } P_b(x) = \sum_{j=0}^{n} b_j x^j$$

durch:

$$c_i = \sum_{l=0}^{i} a_l b_{i-l}$$

Erstellen Sie zuerst ein Struktogramm, aus dem der Aufbau Ihrer Funktion hervorgeht, und codieren Sie die Funktion anschließend in C, wobei auf kurze aber aussagekräftige Kommentare zu achten ist! Auf das Struktogramm entfällt die Hälfte der Punkte!

Aufgabe 2: (20 Punkte)

Schreiben Sie ein Programm **grep**, welches Text, den es zeilenweise von der Standardeingabe einliest, nach dem Vorkommen einer Zeichenkette hin untersucht. Die Zeichenkette wird dabei in der Kommandozeile übergeben, z.B.:

grep Suchbegriff

Wenn die Zeile die angegebene Zeichenkette (hier "Suchbegriff") enthält, wird sie auf die Standardausgabe ausgegeben. Zusätzlich dazu kann das Verhalten des Programms noch durch die folgenden beiden Kommandozeilenoptionen verändert werden:

- **-v** Wird diese Option angegeben, werden im Gegensatz zu vorher diejenigen Zeilen ausgegeben, in denen der Suchbegriff nicht auftritt.
- **-n** Das Programm gibt vor jeder ausgegebenen Zeile zusätzlich die Zeilennummer, in der sie in der Eingabe auftauchte.

Beachten Sie, daß auch beide Optionen gleichzeitig angegeben werden können! Sie können aber zur Vereinfachung davon ausgehen, daß der Suchbegriff als letzter Kommandozeilenparameter angegeben wird. Außerdem kann von einer maximalen Zeilenlänge von 80 Zeichen ausgegangen werden.

Zur Realisierung des Programms stehen Ihnen die Funktionen aus der Zeichenkettenbibliothek zur Verfügung. Unter anderem stellt diese die Funktion

char *strstr(char *string, char *substring)

zur Verfügung. Diese sucht innerhalb der Zeichenkette ***string*** nach der Zeichenkette ***substring***. Enthält ***string*** die andere Zeichenkette, wird die Adresse zurückgegeben, an der ***substring*** innerhalb von ***string*** beginnt, sonst NULL. Um die Funktionen aus dieser Bibliothek verwenden zu können, müssen Sie die Headerdatei **string.h** einbinden!

Wenn das Programm **grep** terminiert, gibt es an den Aufrufer den Wert **0** zurück, falls die Zeichenkette nirgends in der Eingabe gefunden wurde. Sobald mindestens eine Zeile mit der gesuchten Zeichenkette auftrat, wird der Wert **1** zurückgeliefert.

Erstellen Sie auch hier zuerst ein Struktogramm und dokumentieren Sie Ihr Programm mit kurzen, aber aussagekräftigen Kommentaren!

Aufgabe 3: (10 Punkte)

Markieren und verbessern Sie die Fehler (mehr als 10!) in dem untenstehenden Programm, welches von der Kommandozeile zwei ganze Zahlen und ein mathematisches Symbol einliest, die entsprechende Operation auf die beiden Ganzzahlen ausführt und das Ergebnis wieder ausgibt, d.h. etwa: **calc 2 + 3**

#inc lude <stdio .h>
#inc lude <ctype .h>

```
int main(int argc, char argv[])
{

  if(argc!=4) return;

  a = str2int(argv[2]);
  b = str2int(argv[4]);
  switch(*argv[3])   /* Fallunterscheidung für
  {                     die Operatoren */
    case ’+’ : printf("%d + %d = %d\n",a,b,a+b);
               break;
    case ’-’ : printf("%d - %d = %d\n",a,b,a-b);
               break;
    case ’*’ : printf("%d * %d = %d\n";a,b,a*b);
               break;
    case ’/’ : printf("%d / %d = %d\n",a,b,a\b);
    devault :
      printf(unbekannter Operator %s\n,*argv[3]);
  }
  return 0;
}


int str2int(char *string)
{
  int i, n;
  for(i=0; isspace(string{i}), i++);
  sign = (string[i] == '-') : -1 ? 1;
  if((string[i] == '+') || (string[i] = '-')
    i++;
  for(n=0; isdigit(string[i]); i++)
    n = 10 * m + (string[i] - "0");
  return sign*n;
}
```


Aufgabe 4: (10 Punkte)

Was berechnet das folgende Programm? Geben Sie die Inhalte der verwendeten Variablen nach jeder ausgeführten Anweisung an! Verwenden Sie dazu die untenstehende Tabelle!

```
 1   #include <stdio.h>
 2   int k[] = {0,1,0,1,1,0};
 3   int f(int *k, int *v)
 4   {
 5       int m = *(k + *k);
 6       int u,i;
 7
 8       k += 3;
 9       for(u=0,i=0;i<2;i++)
10           u = 2 * u+ *(k++) * m;
11       *v += u;
12   }
13   int main()
14   {
15       int v=0;
16       while(++(*k) < 2) {
17           v *= 2;
18           f(k,&v);
19       }
20       printf("v=%d\n", v);
21   }
```

Zeile							

Aufgabe 5: (10 Punkte)

Gegeben seien die beiden nachfolgenden Programmstücke. Wie oft werden die nachfolgenden Schleifen durchlaufen, und welche Werte werden dabei für die Schleifenvariablen ausgegeben? Um welchen Typ von Schleife handelt es sich jeweils (abweisend ↔ nicht-abweisend)?

a) for-Schleife:

```
int i;
for(i=10;i>5;i--) { printf("%d\n",i);};
```

b) while-Schleife:

```
int i=5;
while(i<12) { i+=2; printf("%d\n",i);}
```

c) do-while-Schleife:

```
int i=2;
do { i+=2; printf("%d\n",i); } while (i<9)
```


Aufgabe 6: (10 Punkte)

Für einen Versandbetrieb soll ein Programm zur Verwaltung von Bestellungen erstellt werden. Als Daten fallen dabei Kundenadressen und Bestellaufträge an.

Entwickeln Sie entsprechende Strukturtypen, wobei sowohl beliebig viele Kundenadressen, wie auch Bestellaufträge möglich sein sollen. Deshalb sollen die Strukturtypen die Speicherung als lineare (verkettete) Liste ermöglichen!

Die **vier** dafür notwendigen Strukturen sollen im einzelnen folgende Daten enthalten:

a) *Kundenadressen* benötigen Felder für Name, Vorname, Straße, Plz., Ort, Kundennummer und eine Liste der Bestellaufträge dieses Kunden.

b) *Bestellaufträge* bestehen aus maximal 20 Bestelleinheiten und einem Datum.

c) *Bestelleinheiten* bestehen aus einer Artikelnummer, der Artikelbezeichnung, der gewünschten Stückzahl und dem Einzelpreis des Artikels.

d) Ein *Datum* besteht aus Tag, Monat und Jahr (nur Zahlen).

Sie können davon ausgehen, daß alle Namen und Bezeichnungen nicht mehr als 20 Zeichen umfassen. Stückzahlen, PLZ u.ä. werden als ganze Zahl und Einzelpreise als Fließkommazahl dargestellt.